#创建部门

CREATE TABLE IF NOT EXISTS dept (

did int not null auto\_increment PRIMARY KEY,

dname VARCHAR(50) not null COMMENT '部门名称'

)ENGINE=INNODB DEFAULT charset utf8;

#添加部门数据

INSERT INTO `dept` VALUES ('1', '教学部');

INSERT INTO `dept` VALUES ('2', '销售部');

INSERT INTO `dept` VALUES ('3', '市场部');

INSERT INTO `dept` VALUES ('4', '人事部');

INSERT INTO `dept` VALUES ('5', '鼓励部');

-- 创建人员

DROP TABLE IF EXISTS `person`;

CREATE TABLE `person` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(50) NOT NULL,

`age` tinyint(4) DEFAULT '0',

`sex` enum('男','女','人妖') NOT NULL DEFAULT '人妖',

`salary` decimal(10,2) NOT NULL DEFAULT '250.00',

`hire\_date` date NOT NULL,

`dept\_id` int(11) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=13 DEFAULT CHARSET=utf8;

-- 添加人员数据

-- 教学部

INSERT INTO `person` VALUES ('1', 'alex', '28', '人妖', '53000.00', '2010-06-21', '1');

INSERT INTO `person` VALUES ('2', 'wupeiqi', '23', '男', '8000.00', '2011-02-21', '1');

INSERT INTO `person` VALUES ('3', 'egon', '30', '男', '6500.00', '2015-06-21', '1');

INSERT INTO `person` VALUES ('4', 'jingnvshen', '18', '女', '6680.00', '2014-06-21', '1');

-- 销售部

INSERT INTO `person` VALUES ('5', '歪歪', '20', '女', '3000.00', '2015-02-21', '2');

INSERT INTO `person` VALUES ('6', '星星', '20', '女', '2000.00', '2018-01-30', '2');

INSERT INTO `person` VALUES ('7', '格格', '20', '女', '2000.00', '2018-02-27', '2');

INSERT INTO `person` VALUES ('8', '周周', '20', '女', '2000.00', '2015-06-21', '2');

-- 市场部

INSERT INTO `person` VALUES ('9', '月月', '21', '女', '4000.00', '2014-07-21', '3');

INSERT INTO `person` VALUES ('10', '安琪', '22', '女', '4000.00', '2015-07-15', '3');

-- 人事部

INSERT INTO `person` VALUES ('11', '周明月', '17', '女', '5000.00', '2014-06-21', '4');

-- 鼓励部

INSERT INTO `person` VALUES ('12', '苍老师', '33', '女', '1000000.00', '2018-02-21', null);

创建数据库内容

1.查询出 教学部 年龄大于20岁,并且工资小于40000的员工,按工资倒序排列.(要求:分别使用多表联合查询和内连接查询)

2.查询每个部门中最高工资和最低工资是多少,显示部门名称

3.求最大工资那个人的姓名和薪水

4.求工资高于所有人员平均工资的人员

5.查询平均年龄在20岁以上的部门名

6.查询教学部 下的员工信息

7.查询大于所有人平均工资的人员的姓名与年龄

-- 查询出 教学部 年龄大于20岁,并且工资小于40000的员工,按工资倒序排列.(要求:分别使用多表联合查询和内连接查询)

select \* from person where age >20 and dept\_id = (select did from dept where dname = '教学部') and salary <40000 ORDER BY salary desc;

-- 查询每个部门中最高工资和最低工资是多少,显示部门名称

select MAX(salary),MIN(salary),dname from person left join dept on person.dept\_id = dept.did GROUP BY dept\_id;

-- 子语句查询

-- 子查询（嵌套查询）：查多次，多个select

-- 注意：第一次的查询结果可以作为第二次的查询的条件或表名使用

-- 注意：临时表的别名不能是char数据类型即表名可以是abcd但不可以是'abcd'

-- 子查询中可以包含：IN, NOT IN ,ANY,ALL.EXISTS和NOT EXISTS

select \* from (select \* from person ) as abcd;

-- 求最大工资那个人的姓名和薪水

select MAX(salary) from person GROUP BY dept\_id;

select \* from person where person.salary IN (select MAX(salary) from person GROUP BY dept\_id);

-- 求工资高于所有人员平均工资的人员

select AVG(salary) from person;

select name from person where salary>(select AVG(salary) from person);

-- 查询平均年龄在20岁以上的部门名

select avg(age),dept\_id from person GROUP BY dept\_id;

select age,dname from (select avg(age)as age,dept\_id from person GROUP BY dept\_id) p left join dept on dept.did = p.dept\_id where p.age >20;

-- 查询教学部 下的员工信息

select did from dept where dname = '教学部';

select \* from person where dept\_id = (select did from dept where dname = '教学部');

-- 查询大于所有人平均工资的人员的姓名与年龄

select AVG(salary) as avg\_salary from person;

select name,age from person where person.salary> (select AVG(salary) from person);

create table liuyan (

id int not null auto\_increment PRIMARY KEY,

title varchar(32) not null,

author varchar(16) null,

addtime datetime not null,

content text not null,

isdelete char(1) not NULL DEFAULT 0

);

select \* from liuyan

alter table liuyan add status char(1) DEFAULT 0; -- 增加字段名 记住对表的操作都是使用alter table

alter table liuyan modify author varchar(16) not null; -- 修改字段属性 记住 字段名后面一定要加数据类型

alter table liuyan alter author set default 'youku'; -- 修改字段属性 添加默认值

alter table liuyan drop isdelete; -- 删除表中的字段

insert into liuyan(id,author,addtime,content,status) values -- 给表插入数据 使用关键字 insert into

(1,'111','2019-01-08 23:53:17','11111',0),

(2,'222','2019-02-02 02:02:02','22222',0),

(3,'333','2019-03-02 02:02:02','33333',0),

(4,'444','2019-04-02 02:02:02','44444',0),

(5,'555','2019-05-02 02:02:02','55555',0);

update liuyan set author = 'admin' where id >3; -- 更新数据使用关键字 update 语法：update 表名 set 要修改的字段名以及参数 where 条件；

select \* from liuyan

delete from liuyan where id =4; -- 删除数据使用 delete from 语法：delete from 表名 where 删除的字段条件

-- 附加题

-- 1、添加大于10条数据 分三个作者添加

insert into liuyan(author,addtime,content,status) values

('aaa','2019-01-01 01:01:01','a1',0),

('aaa','2019-01-01 02:01:01','a2',0),

('aaa','2019-01-01 03:01:01','a3',0),

('aaa','2019-01-01 04:01:01','a4',0),

('bbb','2019-02-01 01:01:01','b1',0),

('bbb','2019-02-01 02:01:01','b2',0),

('bbb','2019-02-01 03:01:01','b3',0),

('bbb','2019-02-01 04:01:01','b4',0),

('ccc','2019-03-01 01:01:01','c1',0),

('ccc','2019-03-01 02:01:01','c2',0),

('ccc','2019-03-01 03:01:01','c3',0),

('ccc','2019-03-01 04:01:01','c4',0);

select \* from liuyan

-- 查询某个作者的留言信息

select content as '内容' from liuyan where author = 'aaa'

-- 查询所有内容 按照时间降序排序

select \* from liuyan order by addtime DESC; -- 排序使用order by 关键字 默认正序asc 倒叙使用 desc

-- 获取id在2到6之间的留言信息，并按时间降序排序

select id,content,addtime FROM liuyan where id>2 and id <6 ORDER BY addtime DESC;

-- 统计每个作者留了多少条留言，并对数量按从小到大排序。-- 使用了聚合函数count 统计非空字段的总数，同时使用了分组查询 group by

select id,author from liuyan GROUP BY author;

select author,count(content) as num from liuyan GROUP BY author order by num;

-- 将id为8、9的两条数据的作者改为’doudou’. 表数据的修改是使用update 表名 set 要修改的字段里的数据

update liuyan set author = 'doudou' where id in (8,9);

select \* from liuyan;

-- 取出最新的三条留言。 -- 通过分页查询

select \* from liuyan ORDER BY addtime desc LIMIT 3;

select \* from liuyan ORDER BY addtime desc ;

-- 查询留言者中包含”a”字母的留言信息，并按留言时间从小到大排序

select \* from liuyan where content like '%a%' ORDER BY addtime;

-- 删除”作者”重复的数据,并保留id最大的一个作者

-- 多表查询的方式--> 先查找出 最大ID的作者--> 接着在将刚刚查询到的数据转为临时表 --> 接着进行 多表查询 使用join 或 where

select \* from liuyan l join (select max(id) as id,author from liuyan group by author) ll on l.author = ll.author where l.id = ll.id

select \* from liuyan where id not in (select id from (select max(id) as id from liuyan group by author) l)

delete from liuyan where id not in (select id from (select max(id) as id from liuyan group by author) l)

-- 查询学习课程"python"比课程 "java" 成绩高的学生的学号;

-- 相当于多表关系 因为 成绩表需要关联学生表和课程表

select c\_name,num,score.s\_id,s\_name from course,score,student where course.c\_id = score.c\_id and course.c\_name = 'python' and score.s\_id = student.s\_id;

select c\_name,num,score.s\_id,s\_name from course,score,student where course.c\_id = score.c\_id and course.c\_name = 'java' and score.s\_id = student.s\_id;

select python.s\_id,python.s\_name from

(select c\_name,num,score.s\_id,s\_name from course,score,student where course.c\_id = score.c\_id and course.c\_name = 'python' and score.s\_id = student.s\_id) as python,

(select c\_name,num,score.s\_id,s\_name from course,score,student where course.c\_id = score.c\_id and course.c\_name = 'java' and score.s\_id = student.s\_id) as java

where python.num >java.num and python.s\_id = java.s\_id;

-- 查询平均成绩大于65分的同学的姓名和平均成绩(保留两位小数);

select ROUND(AVG(num),2) as avg\_num ,score.s\_id,student.s\_name from score,student where score.s\_id = student.s\_id group by s\_id;

select \* from (select ROUND(AVG(num),2) as avg\_num ,score.s\_id,student.s\_name from score,student where score.s\_id = student.s\_id group by s\_id) as msg where msg.avg\_num>65;

-- 查询所有同学的姓名、选课数、总成绩

select \* from student;

select \* from course;

select \* from score order by s\_id;

select COUNT(c\_id),s\_id,sum(num)from score GROUP BY s\_id;

select s\_name as '姓名',count\_c as '选课数',sum\_num as '总成绩' from student,(select COUNT(c\_id) as count\_c,s\_id,sum(num) as sum\_num from score GROUP BY s\_id) as msg

where student.s\_id = msg.s\_id;

-- 查询所有的课程的名称以及对应的任课老师姓名;

select \* from teacher;

select \* from course;

select c\_name,t\_name from course,teacher where course.t\_id = teacher.t\_id;

-- 查询没学过“alex”老师课的同学的姓名;

select \* from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and teacher.t\_name = 'alex';

select s\_name,c\_name,t\_name from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and teacher.t\_name = 'alex';

select s\_name from student where s\_name not in (select s\_name from (select s\_name,c\_name,t\_name from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and teacher.t\_name = 'alex') as msg)

-- 查询学过'python'并且也学过编号'java'课程的同学的姓名;

select c\_name,s\_name from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and course.c\_name = 'python';

select c\_name,s\_name from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and course.c\_name = 'java';

select python.s\_name from (select c\_name,s\_name from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and course.c\_name = 'python') as python inner join (select c\_name,s\_name from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and course.c\_name = 'java') as java on python.s\_name = java.s\_name;

-- 查询学过“alex”老师所教的全部课程的同学的姓名；

select c\_name,student.s\_id from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and teacher.t\_name = 'alex';

select student.s\_name from (select c\_name,student.s\_id as id\_num from course,score,teacher,student where score.c\_id = course.c\_id and course.t\_id = teacher.t\_id and score.s\_id = student.s\_id and teacher.t\_name = 'alex') as msg,student where student.s\_id = msg.id\_num GROUP BY id\_num HAVING count(msg.id\_num)>1;

-- 查询挂科超过两门(包括两门)的学生姓名;

select \* from score where num <60 GROUP BY c\_id ;

-- 查询有超过两门科目（包括两门）及格的学生的姓名（原题目 查询挂科超过两门(包括两门)的学生姓名;）

select score.s\_id as s\_id,c\_id,num ,s\_name from score,student where num >=60 and student.s\_id=score.s\_id;

select s\_name from (select score.s\_id as s\_id,c\_id,num,s\_name from score,student where num >=60 and student.s\_id=score.s\_id) as s GROUP BY s.s\_id HAVING COUNT(s.s\_id)>1;

select \* from score ORDER BY s\_id;

-- 查询有课程成绩小于60分的同学的姓名;

select s\_name from score,student where num<60 and student.s\_id = score.s\_id;

-- 查询选修了课程数大于3的学生姓名;

select score.s\_id,s\_name FROM score,student where student.s\_id = score.s\_id GROUP BY s\_id HAVING COUNT(c\_id)>=3;

select \* from score ORDER BY s\_id;

-- 查询至少有一门课程与“貂蝉”同学所学课程相同的同学姓名；

select \* from student,score where student.s\_id =score.s\_id;

select c\_id,score.s\_id,s\_name from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉';

select s\_name from student,score,(select c\_id,score.s\_id as s\_id from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉') as msg where score.c\_id= msg.c\_id and score.s\_id = student.s\_id and student.s\_name != '貂蝉' group by s\_name;

-- 查询学过'貂蝉'同学全部课程 的其他同学姓名；

select c\_id,score.s\_id,s\_name from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉';

select \* from score order by s\_id;

select s\_name from student,score,(select c\_id,score.s\_id as s\_id from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉') as msg where score.c\_id= msg.c\_id and score.s\_id = student.s\_id and student.s\_name != '貂蝉' group by s\_name HAVING count(msg.c\_id) >=2;

-- 查询和'貂蝉'同学学习的课程完全相同的,其他同学姓名

select s\_id,GROUP\_CONCAT(score.c\_id)as count\_c FROM score,course where score.c\_id = course.c\_id GROUP BY s\_id HAVING count(score.c\_id)= 2;

select \* from score ORDER BY s\_id;

select c\_id,score.s\_id,s\_name,GROUP\_CONCAT(score.c\_id)as count\_c from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉';

select msg.s\_name from

(select c\_id,score.s\_id,s\_name,GROUP\_CONCAT(score.c\_id)as count\_c from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉')as msg,

(select s\_id,GROUP\_CONCAT(score.c\_id)as count\_c FROM score,course where score.c\_id = course.c\_id GROUP BY s\_id HAVING count(score.c\_id)= 2) as p

where msg.count\_c = p.count\_c and s\_name != '貂蝉' GROUP BY msg.s\_id;

-- 按平均成绩倒序显示所有学生的“python”、“java”、“linux”三门的课程成绩，按如下形式显示： 学生ID,python,java,linux,课程数,平均分

select s\_id as '学生ID',

sum(case when c\_id =1 then num else 0 end) as 'python',

sum(case when c\_id =2 then num else 0 end) as 'java',

sum(case when c\_id =3 then num else 0 end) as 'linux',

count(c\_id) as '课程数' ,round(avg(num),2) as '平均分' from score GROUP BY s\_id ORDER BY avg(num) DESC;

select \* from course

select sum(case when c\_id =1 then num else 0 end) as 'python' from score GROUP BY s\_id;

-- 统计各科各分数段人数.显示格式:课程ID,课程名称,[100-85],[85-70],[70-60],[ <60]

select score.c\_id as '课程ID',c\_name as '课程名称',

sum(case when score.num<=100 and num>85 then 1 else 0 end) as'[100-85]',

sum(case when score.num<=85 and num>70 then 1 else 0 end) as'[85-70]',

sum(case when score.num<=70 and num>60 then 1 else 0 end) as'[70-60]',

sum(case when score.num<60 then 1 else 0 end) as'[<60]'

from score,course

where score.c\_id=course.c\_id GROUP BY score.c\_id;

select c\_id,

sum(case when num<=100 and num>85 then 1 else 0 end) as'[100-85]',

sum(case when num<=85 and num>70 then 1 else 0 end) as'[85-70]',

sum(case when num<=70 and num>60 then 1 else 0 end) as'[70-60]',

sum(case when num<60 then 1 else 0 end) as'[<60]'

from score GROUP BY c\_id;

-- 查询每门课程被选修的次数

select c\_id,count(c\_id) from score GROUP BY c\_id;

-- 查询出只选修了一门课程的学生的学号和姓名

select score.s\_id,student.s\_name from score,student where score.s\_id = student.s\_id GROUP BY s\_id HAVING count(c\_id) =1;

-- 查询学生表中男生、女生各有多少人

select sum(case when s\_sex = '男' then 1 else 0 end) as '男',

sum(case when s\_sex = '女' then 1 else 0 end) as '女'

from student;

select \* FROM student;

-- 查询每门课程的平均成绩，结果按平均成绩升序排列，平均成绩相同时，按课程号降序排列

select c\_id, ROUND(avg(num),2)as '平均成绩' FROM score GROUP BY c\_id ORDER BY avg(num),c\_id DESC;

-- 查询课程名称为“python”，且分数低于60的学生姓名和分数

select s\_name,num from course,score,student where course.c\_id =score.c\_id and c\_name = 'python' and student.s\_id = score.s\_id having num < 70 and num >65;

-- 查询和'貂蝉'同学学习的课程完全相同的,其他同学姓名

select s\_id,GROUP\_CONCAT(score.c\_id)as count\_c FROM score,course where score.c\_id = course.c\_id GROUP BY s\_id HAVING count(score.c\_id)= 2;

select \* from score ORDER BY s\_id;

select c\_id,score.s\_id,s\_name,GROUP\_CONCAT(score.c\_id)as count\_c from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉';

select msg.s\_name from

(select c\_id,score.s\_id,s\_name,GROUP\_CONCAT(score.c\_id)as count\_c from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉')as msg,

(select s\_id,GROUP\_CONCAT(score.c\_id)as count\_c FROM score,course where score.c\_id = course.c\_id GROUP BY s\_id HAVING count(score.c\_id)= 2) as p

where msg.count\_c = p.count\_c and s\_name != '貂蝉' GROUP BY msg.s\_id;

-- 查询和'貂蝉'同学学习的课程完全相同的,其他同学姓名

select s\_id,GROUP\_CONCAT(score.c\_id)as count\_c FROM score,course where score.c\_id = course.c\_id GROUP BY s\_id HAVING count(score.c\_id)= 2;select GROUP\_CONCAT(score.c\_id)as count\_c,s\_name from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉';

select msg.s\_name from

(select GROUP\_CONCAT(score.c\_id)as count\_c,s\_name from student,score where student.s\_id =score.s\_id and student.s\_name= '貂蝉')as msg,

(select s\_id,GROUP\_CONCAT(score.c\_id)as count\_c FROM score,course where score.c\_id = course.c\_id GROUP BY s\_id HAVING count(score.c\_id)= 2) as p

where msg.count\_c = p.count\_c and s\_name != '貂蝉' GROUP BY p.s\_id;